

---

# RoCKIn@Work: Industrial Robot Challenge

---

Rainer Bischoff, Tim Friedrich,  
Gerhard K. Kraetzschmar, Sven Schneider and  
Nico Hochgeschwender

Additional information is available at the end of the chapter

<http://dx.doi.org/10.5772/intechopen.70014>

---

## Abstract

RoCKIn@Work was focused on benchmarks in the domain of industrial robots. Both task and functionality benchmarks were derived from real world applications. All of them were part of a bigger user story painting the picture of a scaled down real world factory scenario. Elements used to build the testbed were chosen from common materials in modern manufacturing environments. Networked devices, machines controllable through a central software component, were also part of the testbed and introduced a dynamic component to the task benchmarks. Strict guidelines on data logging were imposed on participating teams to ensure gathered data could be automatically evaluated. This also had the positive effect that teams were made aware of the importance of data logging, not only during a competition but also during research as useful utility in their own laboratory. Tasks and functionality benchmarks are explained in detail, starting with their use case in industry, further detailing their execution and providing information on scoring and ranking mechanisms for the specific benchmark.

**Keywords:** robotics, robot competitions, benchmarking, domestic robots, industrial robots

---

## 1. Introduction

RoCKIn@Work is a competition that aims at bringing together the benefits of scientific benchmarking with the economic potential of innovative robot applications for industry, which call for robots capable of working interactively with humans and requiring reduced initial programming.

The following user story is the basis on which the RoCKIn@Work competition is built: RoCKIn@Work is set in the RoCKIn'N'RoLLIn factory—a medium-sized factory that is trying

to optimize its production process to meet the increasing number of unique demands from its customers. RoCKIn'N'RoLLIn specializes in the production of small- to medium-sized lots of mechanical parts and assembled mechatronic products. Furthermore, the RoCKIn'N'RoLLIn production line integrates incoming shipments of damaged or unwanted products and raw materials. A key requirement to ensure the competitiveness of European industry is greater automation in a wide range of application domains which include flexible production processes that can easily be adapted to customer demands.

In RoCKIn@Work, robots will assist with the assembly of a drive axle—a key component of the robot itself and therefore a step towards self-replicating robots. Tasks include locating, transporting and assembling necessary parts, checking their quality and preparing them for other machines and workers. By combining the versatility of human workers and the accuracy, reliability and robustness of mobile robot assistants, the entire production process is able to be optimized.

RoCKIn@Work is looking to make these innovative and flexible manufacturing systems, such as that required by the RoCKIn'N'RoLLIn factory, a reality. This is the inspiration behind the challenge and the following scenario description.

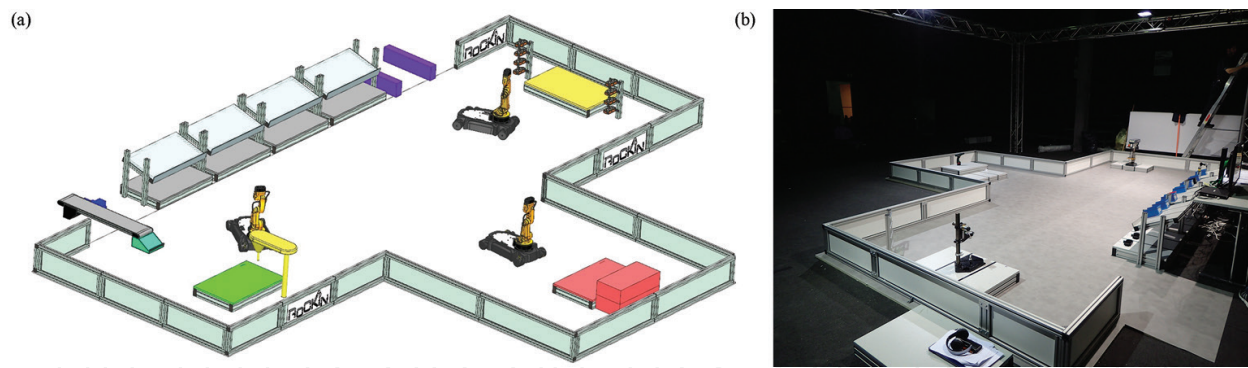
Section 2 gives an oversight of the RoCKIn'N'RoLLIn factory and introduces all hardware and software elements that were used. Section 3 gives a detailed description of the task benchmarks, the way they have to be executed and the way they are scored. It further gives an explanation on the decisions that were taken to create the task benchmarks and how they differ from other benchmarks. Section 4 does the same for the functional benchmarks. The last section of this chapter gives a short summary and details some of the impressions from RoCKIn camps and competitions.

## 2. The RoCKIn@Work environment

This section introduces all hardware and software elements that are needed for the RoCKIn'N'RoLLIn factory to come to life. The description focuses on the elements themselves. A more detailed overview, especially on the software infrastructure, is given in Ref. [1].

### 2.1. The RoCKIn@Work testbed

The testbed for RoCKIn@Work, explained in detail in Ref. [2], consists of the environment in which the competition took place, including all the objects and artefacts in the environment, and the equipment brought into the environment for benchmarking purposes. An aspect that was comparatively new in robot competitions is that RoCKIn@Work is, to the best of our knowledge, the first industry-oriented robot competition targeting an environment with ambient intelligence, i.e. the environment is equipped with networked electronic devices the robot can communicate and interact with, which allow the robot to exert control on certain environment artefacts like conveyor belts or machines. **Figures 1a** and **b** show the testbed as it was used during the RoCKIn@Work competition 2015 in Lisbon.



**Figure 1.** The RoCKIn@Work testbed. (a) Planned setup of the testbed and (b) Actual testbed at RoCKIn competition 2015.

## 2.2. Environment elements

To create an environment that closely resembles a real factory shop floor, a lot of different elements are necessary. In the case of RoCKIn@Work, the following elements are used:

- A set of shelves
- A force fitting workstation
- A drilling workstation
- A conveyor belt with a quality control camera mounted to it
- An assembly workstation
- A set of objects to be manipulated

**Figure 2** shows an overview of the testbed elements.

Shelves are used to place objects. These objects range from a single object, e.g. a bearing, to containers storing multiple objects at once. The containers are so-called small load carriers. They are standardized containers in industry, originally meant to optimize the logistics chain of the automotive industry and their suppliers. The set of shelves area in RoCKIn@Work is a set of connected shelves and each shelf has two levels (upper level and lower level, see also **Figure 2**, lower right corner). The robot takes and/or delivers objects from the shelves (through the containers or directly onto shelves). The shelves are built from metal profiles and wooden panels, also something common on every factory floor. To make transportation and set-up easy, the construction of the shelves follows a modular design. Set-up and dismantling of all components can be done using a single Allen key. All components of the testbed fit on a single euro pallet after dismantling.

The force fitting workstation consists of a table for temporarily storing handled parts. The table itself is part of the force fitting machine, which can be operated by a robot or human worker. For this purpose, a drive was fixed to its structure. The drive is connected to a control board, which is attached to a Raspberry Pi microcontroller board running the necessary



**Figure 2.** Elements of the RoCKIn@Work testbed.

software to control the drive. On one side of the force fitting workstation, an assembly aid tray rack is placed. This assembly aid tray rack can be used to attach filled or unfilled aid trays, 3D-printed containers that can hold up to two bearing boxes, or finished assemblies. A more detailed description is given later in this section.

The drilling workstation consists of a storage area to store *file card* boxes and the drilling machine. The drilling machine is a simple model that can be purchased at a hardware store. Like for the force fitting machine, a drive with control board and a Raspberry Pi board were fixed to it so that the upwards/downwards motion can be controlled by a computer. Next to it, a conveyor belt is placed.

The conveyor belt transports parts from outside of the arena into the area. At the end of the conveyor belt, a quality control camera (QCC) was mounted. The camera is connected to the testbed's network and able to communicate with the robot through the Central Factory Hub (CFH; detailed below). Parts delivered into the arena fall down through guiders on an exit ramp in a predefined position where they can be taken by the robot.



The assembly workstation consists of a table, where a human worker can perform assembly of parts. The table features predefined areas where the robot can put boxes with supplies and pick up boxes with finished parts that had already been processed by the worker and need to be delivered elsewhere [3].

The objects present in the testbed can be subdivided into three classes as follows:

- Mechanical parts that have to be recognized and manipulated.
- Objects in the environment that have to be recognized and manipulated.
- Objects in the environment that have to be recognized only (because they are fixed to the environment, too heavy for the robot to lift or not relevant for the task).

**Figure 3** shows the objects available in the testbed.

### 2.3. Central Factory Hub (CFH)

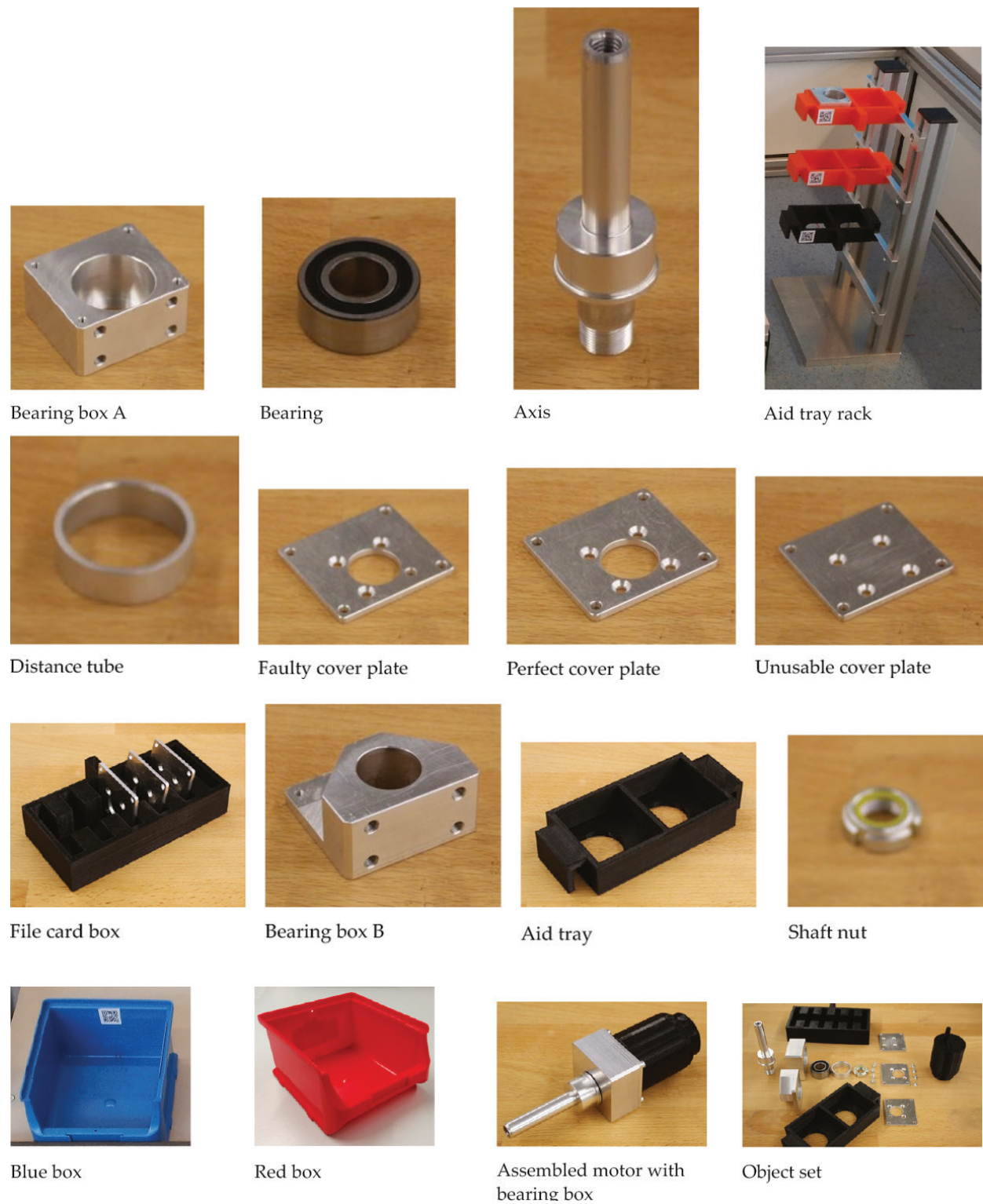
The main idea of the RoCKIn@Work testbed software infrastructure is to have a central server-like hub (the RoCKIn@Work Central Factory Hub) that serves all the services that are needed for executing and scoring tasks and successfully realizing the competition. This hub was derived from software systems, which are well known in industrial business (e.g. SAP). It provides the robots with information regarding the specific tasks and tracks the production process as well as stock and logistics information of the RoCKIn'N'RoLLIn factory. It uses a plug-in software architecture. Each plug-in is responsible for a specific task, for benchmarking or for other functionalities. A detailed description of the CFH and how it is utilized during RoCKIn and other robot competitions is given in Ref. [1].

### 2.4. Networked devices in the environment

The four networked devices described previously are used during execution of task benchmarks. This paragraph provides an overview on the capabilities of each networked device and its role in the related task. All networked devices can be operated through their connection to the Central Factory Hub. The software interface allows control either by the robot or through a graphical user interface by a human operator.

The force fitting machine is used for the insertion of a bearing into a bearing box. The force fitting process is performed by first inserting a bearing box with bearing on top of the bearing box. The placement process is executed with the help of an assembly aid tray. After the bearing box and bearing are properly placed, the force fitting machine is instructed to move down. Finally, the force fitting machine is instructed to move up again to make pick up of the processed item possible. The force fitting machine is used in the *Prepare Assembly Aid Tray for Force Fitting* task.

The drilling machine is used for drilling a cone sink in a cover plate. It is equipped with a customized fixture for the plates. Like for the force fitting machine, the drilling machine can be operated through its network interface with the CFH. The robot first has to insert the cover



**Figure 3.** Objects in the RoCKIn@Work testbed.

plate into the fixture of the drilling machine. After that, the robot signals to the CFH to move the drill head down. Finally, the drill is moved up again and the drilled cover plate can be picked up. The drilling machine is used in the *Plate Drilling* task, specifically for the correction of a faulty cover plate.

The conveyor belt is used for delivering parts into the RoCKIn@Work testbed. At its end, it has a quality control camera to detect defects on the parts which are being delivered. The conveyor belt can be commanded, by the quality control camera, to move in both directions and to start/stop. It is not possible for the robot to directly interface with it. The conveyor belt is used in the *Plate Drilling* task.

The quality control camera or QCC is mounted above the conveyor belt and is used to acquire information about the quality of incoming cover plates delivered through the conveyor belt. The QCC also has the responsibility to deliver only a single cover plate through the conveyor belt (until the cover plate reaches the exit ramp of the conveyor belt) for each received command. After receiving a command, the QCC activates the conveyor belt until a cover plate is within the viewing range of the QCC. At this point, the QCC detects any defects of the cover plate. The conveyor belt keeps moving until it is being stopped by the QCC when the cover plate reaches the exit ramp of the conveyor belt. The QCC is used for the *Plate Drilling* task.

## 2.5. Benchmarking equipment in the environment

RoCKIn benchmarking is based on the processing of data collected in two ways as follows [3]:

- **Internal benchmarking data:** collected by the robot system under test.
- **External benchmarking data:** collected by the equipment embedded into the testbed.

External benchmarking data are generated by the RoCKIn testbed using a multitude of methods, depending on the nature of the data. One type of external benchmarking data used by RoCKIn is pose data about robots and/or their constituent parts. To acquire these, RoCKIn uses a camera-based commercial motion capture system (MCS) composed of dedicated hardware and software. Benchmarking data have the form of a time series of poses of rigid elements of the robot (such as the base or the wrist). Once generated by the MCS, pose data is acquired and logged by a customized external software system based on Robot Operating System (ROS). More precisely, logged data is saved as bagfiles created with the rosbag utility provided by ROS. Pose data is especially relevant because it is used for multiple benchmarks. There are other types of external benchmarking data that RoCKIn acquired. However, these are usually collected using devices specific to the benchmark. Finally, equipment to collect external benchmarking data includes any server which is part of the testbed and which the robot subjected to a benchmark has to access as part of the benchmark. Communication between servers and robot is performed via the testbeds' own wireless network. An extensive analysis on evaluation criteria and metrics for benchmarking is given in Ref. [4].

## 3. Task benchmarks

The concept of task benchmarks has already been introduced in Chapter 1. This section therefore describes details concerning rules, procedures, as well as scoring and benchmarking methods, which are common to all task benchmarks in the RoCKIn@Work competition.

To make repeatability and reproducibility of the task benchmarks possible, teams have to follow a set of rules which are meant to lead to a more scientific benchmarking approach [5] instead of simply ‘hacking’ to get around a problem. To ensure a safe competition both for teams as well as the audience, every run of each of the task benchmarks has been preceded by a safety check. This is a very important aspect that often, especially with younger students, does not get sufficient attention. Much more often, a quick solution to a problem is found, but at the risk of injury. To avoid potential damage to the testbed or injury to participants, the team members must ensure and inform at least one of the organizing committee (OC) members present during the execution of the task that they have an emergency stop button on the robot which is fully functional. Any member of the OC can ask the team to stop their robot at any time, and such requests must be honoured immediately and swiftly. The OC member present during the execution of the task also makes sure that the robot is compliant with all safety-related rules and robot specifications defined in the rulebook. All teams are required to perform each task according to the steps mentioned in the ‘Rules and Procedures’ subsections for the tasks. During the competition, all teams are required to repeat a task benchmark multiple times. Each benchmark run is limited by a specified period of time.

During RoCKIn, benchmarking is of great importance. To gather as much information as possible and process the information later without error, guidelines on data storage had to be followed. This list presents the guidelines that are common to all task benchmarks. Specific information that has to be logged, but that only occurred during a single benchmark, is given later on in the description of the specific task benchmark.

- **Calibration parameters:** The calibration parameters for cameras have to be saved. This must also be done for other sensors that require calibration (e.g. Kinect), if a calibration procedure is applied instead of using the default values (e.g. those provided by OpenNI).
- **Notes on data storage:** The specific data that the robot has to save are described in the benchmark section. In general, some data streams (those with the highest bitrates) have to be logged only at time intervals when they are actually used by the robot to perform the activities required by the benchmark [3]. Thereby, system load and data bulk can be minimized. For instance, whenever a benchmark includes object recognition activities, video and point cloud data have to be logged by the robot only at times when performing object recognition.
- **Use of data:** The logged data is not used during the competition, in particular, it is not used for scoring. RoCKIn processed the data after the end of the competition. It was used for in-depth analysis and/or to produce datasets, published online, as given in ref. [3], for the benefit of the robotics community.
- **Where and when to store logged data:** Robots are required to store logged data in a specified format on an USB stick. The USB stick is given to the team immediately before the start of the benchmark by one of the RoCKIn partners and has to be returned (with the required data on it) at the end of the benchmark. All files produced by the robot that are associated with the execution of the benchmark have to be handed over.

Since benchmarking and data logging during robot competitions was a new concept, most teams were unaware of the implications this had on their system. To make sure that the data



gathered lead to accurate results, teams were trained during the RoCKIn Camps and the RoCKIn Field Exercise on the principles of data logging. The camp and the field exercise usually took place early in the competition year, whereas the competition was held during the end of a year. The hands-on experience and help by the RoCKIn experts during the camp/field exercise led to most teams being able to correctly log the required data. Overall team performance was increased a lot and problems with the use of the software infrastructure provided by RoCKIn were minimized.

During the competitions, evaluation of the performance of a robot according to its task benchmark description is based on performance equivalence classes and they are related to whether the robot has performed the required task or not. The criterion determining the performance equivalence class of a robot is based on the concept of tasks requiring achievements, while the ranking of robots within each equivalence class is obtained by looking at the performance criteria. Specifically, the performance of a robot belonging to performance class  $N$  is considered better than the performance of a robot belonging to performance class  $M$  whenever  $M < N$ . In case, two robots fall into the same performance class, then a penalization criterion is used (penalties are defined according to task performance criteria) and the performance of the robot which received fewer penalizations is considered better. Finally, if two robots received the same number of penalizations, the performance of the robot which finished the task more quickly is considered better (unless not being able to reach a given achievement within a given time was already explicitly considered as a penalty). Thus, performance equivalence classes and in-class ranking of the robots are determined according to three sets as follows [3]:

- A set  $A$  of achievements, i.e. things that should happen (what the robot was expected to do).
- A set  $PB$  of penalized behaviours, i.e. robot behaviours that are penalized, if they happen, (e.g. bumping into the testbed).
- A set  $DB$  of disqualifying behaviours, i.e. robot behaviours that absolutely must not happen (e.g. hitting people).

Scoring was implemented with the following three-step sorting algorithm:

- If one or more of the elements of set  $DB$  occurred during task execution, the robot gets disqualified (i.e. assigned to the lowest possible performance class, called class 0), and no further scoring procedures are performed.
- Performance equivalence class  $X$  is assigned to the robot, when  $X$  corresponds to the number of achievements in set  $A$  that the robot accomplished.
- Whenever an element of set  $PB$  occurred, a penalty is assigned to the robot (without changing its performance class).

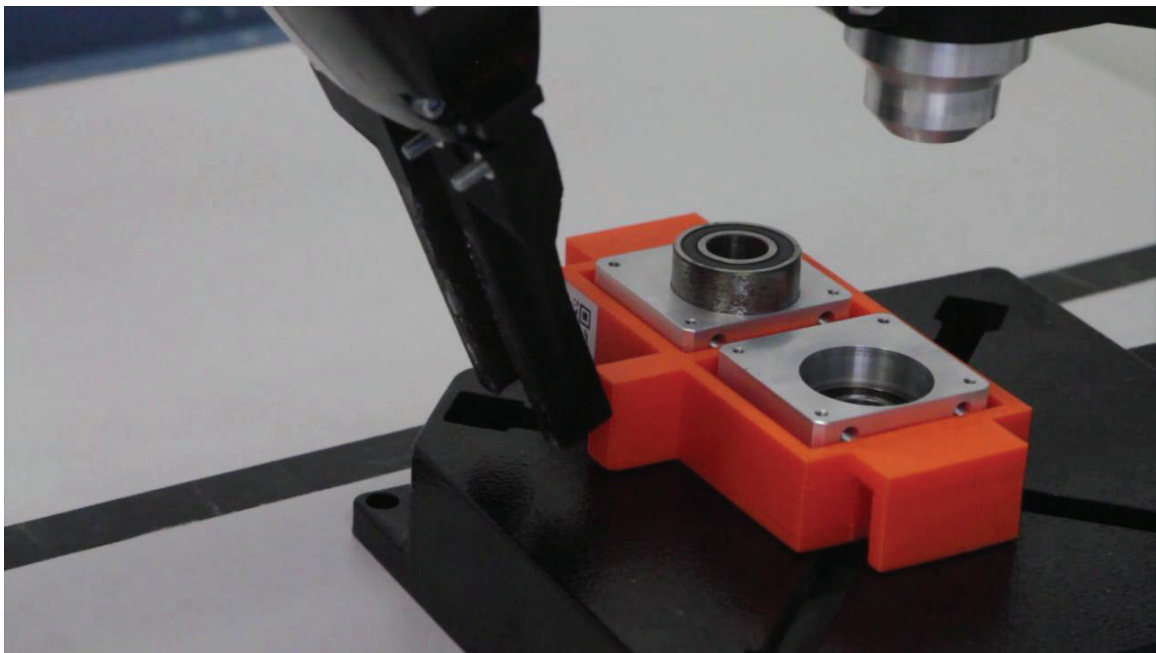
One key property of this scoring system is that a robot that executed the required task completely is always placed into a higher performance class than a robot that executed the task only partially. Moreover, penalties do not cause a change of class (also in the case of incomplete task execution).

### 3.1. Prepare assembly aid tray for force fitting

This task serves as an example for collecting and assembling parts from different locations. Additionally, teams can show their robot’s capability in loading and unloading machines (a well-known industrial task). At the side of the aid tray—a container specifically built to hold two bearing boxes—unique identifiers are attached to uniquely identify the object. This task links to the concept of human robot collaboration (HRC), an idea that becomes more and more import in future factory environments. In this scenario, robots are not meant to take over human workers’ jobs, but to support them and assist them with repetitive tasks or possibly unhealthy activities. This will be increasingly important in the future to react to increasing demand for customized products and to meet market demands.

#### 3.1.1. Task description

The robot’s task is to collect bearing boxes from stock (shelves) and to insert them into specialized aid trays. It is expected that the robot moves to the central station and registers with the Central Factory Hub. After receiving the task of *Assembly Aid Tray for Force Fitting*, the robot should locate the assembly aid tray in the shelf and proceed with identifying the identifiers on the assembly aid tray. The identifier encodes information like the assembly aid tray’s serial number and the type of bearing box which can be fitted. Based on the examination of the assembly aid tray, the robot needs to find the correct bearing boxes in the shelves area. After finding the right bearing boxes, the robot has to record the identifiers of their containers, collect the bearing boxes and place them into the assembly aid tray. It can choose whether to deliver the bearing boxes collectively or individually based on its own reasoning. Once the assembly aid tray is filled with the bearing boxes, the trays can be loaded onto a force fitting machine, where the bearings are force fitted into the bearing boxes (see **Figure 4**).



**Figure 4.** Force fitting of bearing into bearing box.

When these steps of the process are completed, the robot gets a confirmation from the Central Factory Hub and has to perform a final examination of the finished product before its delivery. By scanning the identifiers as part of the task, the robot ensures continuous tracking of the production process and the parts involved. To make the challenge more realistic, some feature variation is possible. For example, the bearing boxes may come in different shapes. This variation is motivated by the modular concept of the final product, where the bearing box has to be inserted in different chassis. The robots are allowed to collect and insert the bearing boxes in the assembly aid tray individually or collectively.

### *3.1.2. Procedures and rules*

Teams are provided with the following information:

- Description of the set of possible assembly aid trays and bearing boxes.
- Description and location(s) of the container(s) used for the bearing boxes.

During the task execution, the robot should perform the task autonomously and without any additional input. The task benchmark has to be carried out following these steps:

1. The robot is provided with multiple assembly aid trays and information about where the bearing boxes are stored.
2. Based on the identifier provided to the teams beforehand, the robot has to identify the appropriate bearing boxes it needs to put on the tray.
3. From the storage area, the robot has to pick the bearing boxes identified in Step 2 and insert them into the provided assembly aid tray.
4. The robot has to deliver the assembly aid tray to the force fitting workstation to be processed further.

Teams also have to be aware that an additional robot may be randomly moving in the arena which must be avoided by the participating robot. Although this randomizing element had been a possible feature variation, it was never actually applied in past competitions, because the dynamics of this feature could have had a negative impact on the repeatability and reproducibility of this benchmark.

### *3.1.3. Scoring and ranking*

The performance equivalence classes used to evaluate the performance of a robot in this task benchmark are defined in relation to four different task elements. The first class is based on whether the robot correctly identified the assembly aid tray or not. The second class is defined by the robot correctly identifying the container or not. Class three uses the number of bearing boxes successfully inserted into the aid tray, and class four rewards the successful execution of the force fitting procedure. The fourth class encourages teams to try and solve the complete task instead of focusing on scoring only through pick-and-place actions.

The complete set  $A$  of achievements in this task included

- Correct identification of the assembly aid trays identifier
- Correct identification of the containers identifier
- Correct grasp of the assembly aid tray
- Correct grasp of the first bearing box
- Correct grasp of the second bearing box
- Correct insertion of the first bearing box into the aid tray
- Correct insertion of the second bearing box into the aid tray
- Correct delivery of the tray to the force fitting station
- Completely processing a part (from identifying to delivering)
- Cooperating with the CFH and networked devices throughout the task

At the end of the task benchmark, the team has to deliver the benchmarking data logged on a USB stick to one of the RoCKIn partners. If delivered appropriately and according to the guidelines, the team can score an additional achievement.

During the run, the robot should not bump into obstacles in the testbed, drop any object previously grasped or stop working. These behaviours are considered as behaviours that need to be penalized, and hence they are added to set  $PB$  of penalized behaviours.

The robot can demonstrate various behaviours that can lead to its disqualification. This includes, for instance, (a) if it damages or destroys the objects to be manipulated or the testbed; (b) if it shows extremely risky or dangerous behaviour; or (c) its emergency stop button is dysfunctional. Such disqualifying behaviours are added to the set  $DB$ .

### 3.2. Plate drilling

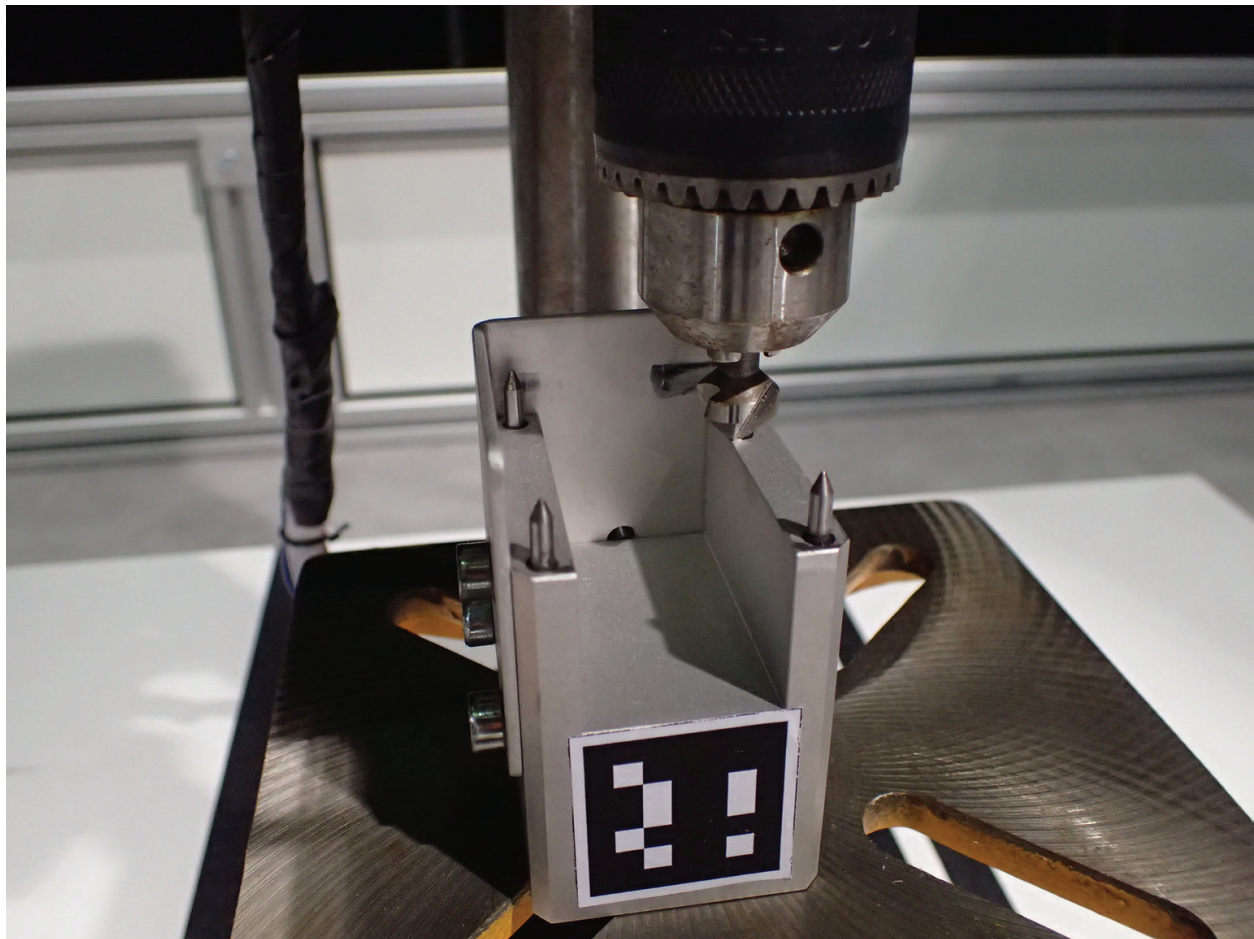
This task simulates handling of incomplete or faulty parts received from an external component supplier. The factory has to quickly react in such cases and create a process to correct the faulty parts. In principle, this task very closely corresponds to a real-world application. Not being able to manufacture components due to faulty incoming supplies can very quickly cost a lot of money. Especially, in times of 'just-in-time-manufacturing', where only small numbers of components are in stock, a faulty delivery can lead to a standstill of large parts of the production line. Being able to react fast and solve smaller issues yourself is crucial for manufacturing.

#### 3.2.1. Task description

The cover plate of the bearing box has eight holes for connecting the motor with the bearing box. The four central holes need to have a cone sink. There are two possible defects of a cover



plate which need to be accommodated in this task. The first case is where the supplier forgot to drill one of the cone sinks which results in a faulty cover plate. The faulty cover plates can be corrected by drilling the cone sink with the drilling machine available in the factory. The second case is where the cover plate is unusable and needs to be returned to the supplier for replacement. Examples of perfect, faulty and unusable cover plates are shown in **Figure 3**. The benchmark starts when the robot receives the task from the CFH. While performing the task, the robot has control over the QCC which allows it to regulate the flow of incoming cover plates. The robot has to send a command to the CFH to operate the QCC. Once the QCC receives a command from the CFH, the QCC activates the conveyor belt until a cover plate is placed on the exit ramp of the conveyor belt. During this process, the QCC detects the type of cover plate which is being delivered (either faulty or unusable) and sends this information to the CFH. Finally, the CFH broadcasts this information so that the robot knows that a faulty or an unusable cover plate was placed on the exit ramp of the conveyor belt. For each cover plate that arrives in the conveyor belt exit ramp, the robot needs to process them according to their fault status. An unusable cover plate needs to be delivered to the trash container box in the factory. For a faulty cover plate, the robot needs to perform correction by delivering it to the drilling machine (see **Figure 5**), operating the drilling machine to fix the missing cone sink, and placing the corrected plate in the file card box.



**Figure 5.** Retainer for faulty cover plate placement.

This benchmark also provides some possibilities for feature variation. For example, the sequence of faulty, unusable and perfect cover plates flowing over the conveyor belt is not fixed. This becomes relevant for the way a robot has to pick up cover plates from the exit ramp. If the robot needs to place the cover plate in the drilling machine for rework, specific positioning of its gripper may be required, while grip position is less important for the unusable plates, which end up in the trash container. The same holds for the orientation of the cover plate on the conveyor belt. In the first competition, it was planned that plate orientation is random, which would have led to more possibilities of grasping the plate from the exit ramp. For the second competition, this variation was not permitted in the spirit of repeatability. This is also true for the last variation. The number of plates delivered in each category (faulty, unusable and perfect) should have been randomized in each benchmark run, but it was decided that all teams should be able to execute exactly the same test. Furthermore, the solutions can vary depending on the sequence of activities being performed by the robot. The robot can choose to collect all cover plates from the conveyor belt first and process them collectively or perform the task for one cover plate at a time before collecting the next cover plate from the conveyor belt. The many variations possible through the robot's own reasoning and performance make the task benchmark challenging enough that none of the other variations were actually applied so far. The focus was instead set on repeatability, reproducibility and fairness between benchmark runs and teams.

### 3.2.2. Procedures and rules

Teams are provided with the following information:

- 3D CAD-textured models of the plates.
- Description of three different states of the plate (faulty, unusable and perfect).
- Location of objects related to the task.
- Commands for operating the QCC and the drilling machine.

During execution of the task, the robot should perform the task autonomously and without any additional input. The task benchmark is carried out by executing the following steps:

1. The robot controls the QCC until it receives feedback that a cover plate has arrived in the exit ramp of the conveyor belt.
2. The robot should pick up the cover plate and proceed with processing the cover plate as described in the next step.

According to the three possible fault types of the cover plate received, there are three possible (sequences of) actions to be executed by the robot as follows:

- a. If the cover plate is perfect, the robot should place it in the file card box.
- b. If the cover plate is unusable, the robot should drop it into the trash container box.

- c. If the cover plate is faulty, the robot should place the cover plate into the drilling machine, then perform the correction of the cover plate using the drilling machine, and finally place the corrected cover plate in the file card box.

In this benchmark, similar to the *Prepare Assembly Aid Tray for Force Fitting*, teams also have to be aware that an additional robot may be randomly moving in the arena. For the same reasons as mentioned in the preceding benchmark, this variation element was not yet applied during the benchmarking exercises and competitions.

### 3.2.3. Scoring and ranking

The performance equivalence classes used to evaluate the performance of a robot in this task benchmark are defined in dependence to three different categories. The first category is based on the number and percentage of correctly processed faulty cover plates. The second class refers to the number and percentage of correctly processed unusable cover plates. The third class uses only the execution time as measure (if less than the maximum time allowed for the benchmark was used by the robot). To encourage teams to try and solve the complete task in the *Plate Drilling* benchmark, it is also possible to score 'extra' achievements for the completion of a whole task (from request to delivery of a cover plate).

The complete set A of possible achievements in this task includes successful execution of

- Communication with the CFH throughout the test.
- Picking up a cover plate from the conveyor belt exit ramp.
- Placing an unusable cover plate in the trash container box.
- Complete handling an unusable cover plate (picking up an unusable cover plate from the conveyor belt exit ramp and placing it in the trash container box).
- Placing a faulty cover plate into the drilling machine.
- Performing the drilling of a faulty cover plate using the drilling machine.
- Complete handling a faulty cover plate (picking up a faulty cover plate from the conveyor belt exit ramp, placing it into the drilling machine, and performing the drilling of the faulty plate using the drilling machine).
- Picking up a corrected cover plate from the drilling machine.
- Placing a corrected cover plate into the file card box.
- Complete handling of a corrected cover plate (picking up a corrected cover plate from the drilling machine and placing it into the file card box).

At the end of the task benchmark, the team has to deliver the benchmarking data logged on a USB stick to one of the RoCKIn partners. If delivered appropriately and according to the guidelines, the team can score an additional achievement.

During the run, the robot should not bump into obstacles in the testbed, drop any object previously grasped or stop working. These behaviours are considered as behaviours that need to be penalized, and hence they are added to set *PB* of penalized behaviours.

The same disqualifying behaviours as in task benchmark *Plate Drilling* do apply for this task benchmark.

### 3.3. Fill a box with parts for manual assembly

This task benchmark reflects one of the primary requirements for a mobile robotic service assistant working together with humans. It is one of the most common tasks in industry: transporting parts from stock to the shop floor or to a human worker is very time consuming and requires well-planned logistics processes. For a human, it is cumbersome to check during his tour, if anything has changed or if he could pick up additional parts on his way. An automatic system has the advantage of direct communication to the shop floor management system and it can quickly respond and replan, if anything changes during production. The human worker can focus on his assembly task instead of worrying about parts arriving on time. This summarizes the idea behind this benchmark. The goal is to assist humans at a manual assembly workstation by delivering parts from different shelves to a common target location.

#### 3.3.1. Task description

The robot has to fill boxes with parts for the final manual assembly of a drive axle. The task execution is triggered by the robot receiving a list of parts required for the assembly process from the CFH. It then proceeds by first collecting an empty box from the shelves, then collecting the requested parts (individually or collectively). When the parts have been placed in the box (see **Figure 6**), the robot delivers the box to the assembly workstation and provides the human worker with a list of parts in the box and a list of missing parts, if any. The boxes have no specific subdivisions; they may have foam material at the bottom to guarantee the safe transport. Thus, the robot has to plan the order of collecting the parts so that they can be easily arranged next to each other.

Feature variation in this task is kept to a minimum. Since it is a common task in industry, possible variations include different boxes, different parts and different locations for the parts. The planning and scheduling to best process the order is left to the teams. The benchmark itself aims for teams to show a good performance. All functional components of a robot system have to be used to solve the task, including navigation, object recognition, planning and manipulation. The benchmark still allows for more errors than the other benchmarks, e.g. position and orientation accuracy during navigation.

#### 3.3.2. Procedures and rules

Teams are provided with the following information:

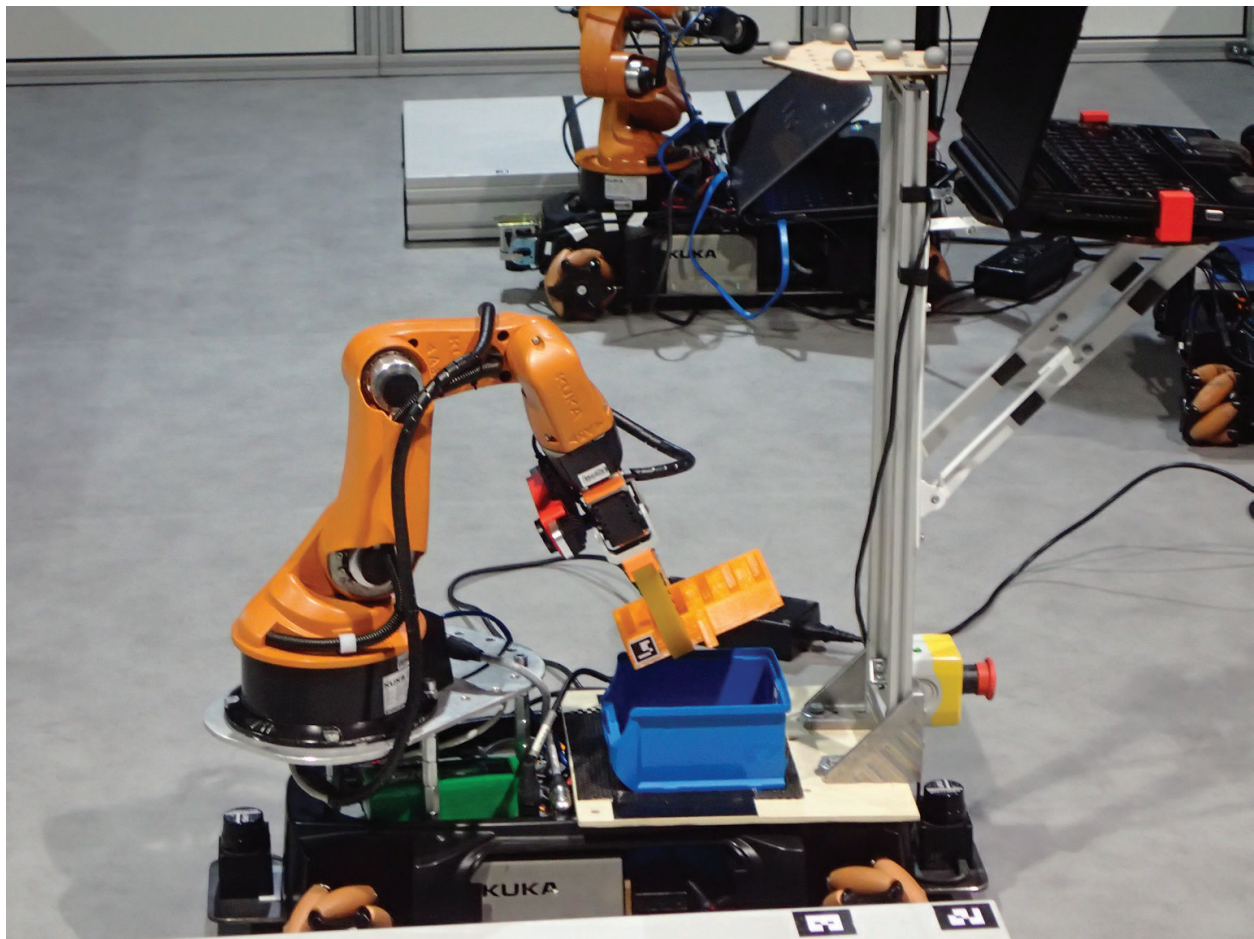
- The list of possible parts used in the task.



- Description of the box used for collecting the parts.
- Location of the parts in the arena.

During the execution of the task, the robot should perform the task autonomously and without any additional input. The task benchmark is carried out by executing the following steps:

1. The robot receives an order for a final assembly step of a product from the CFH containing a list of objects to be collected and delivered.
2. The robot plans the best path to the designated workstation, passing through each storage area where the required objects can be found.
3. The robot must move along the above path, collect the objects and deliver them to the designated area for final product assembly.
4. The Steps 2 and 3 above have to be performed for all the products in the list given in Step 1. Also, the robot has to follow, as much as possible, the priorities imposed by the philosophy of first-in/first-out when executing Steps 2 and 3.



**Figure 6.** Placing an assembly aid tray into small load container.

There may be multiple obstacles present in the scene that may block the direct path planned by the competing robot. If this is the case, the robot has to avoid all the obstacles or other robots during the execution of its task. To keep the benchmark repeatable and fair, new obstacles introduced to the testbed were positioned in the same place for all teams.

### 3.3.3. Scoring and ranking

The performance equivalence classes used to evaluate the performance of a robot in this task benchmark are defined in dependence to two different categories. The first class relates to the number of parts actually provided by the robot to the human worker, and the second class is based on how well the order of arrival corresponds to the desired one.

The complete set  $A$  of possible achievements in this task includes

- Communication with the CFH throughout the test.
- Picking up a required object (also the container) from its storage location.
- Placing the required objects into the container.
- Delivering a correctly filled container to the designated workstation.

At the end of the task benchmark, the team has to deliver the benchmarking data logged on a USB stick to one of the RoCKIn partners. If delivered appropriately and according to the guidelines, the team can score an additional achievement.

During the run, the robot should not bump into obstacles in the testbed, drop any object previously grasped or stop working. These behaviours are considered as behaviours that need to be penalized, and hence they are added to the set  $PB$  of penalized behaviours.

In this task benchmark the same disqualifying behaviours as in the previously mentioned task benchmarks are considered.

## 4. Functionality benchmarks

The concept of functionality benchmarks has already been introduced in Chapter 1. This section therefore describes details concerning rules, procedures, as well as scoring and benchmarking methods, which were common to all functional benchmarks in the RoCKIn@Work competition.

The basic execution and data logging guidelines as explained in Section 3 are also applied for functional benchmarks. Since communication with the CFH is of more importance in the functional than in the task benchmarks, teams need to follow additional rules. The easiest rule for the robot is to send a *BeaconSignal* message at least every second. This ensures that the CFH can detect whether a robot is still working or not. This also makes it possible to track when and how long a robot may have lost the connection to the CFH, for example, due to problems with the wireless network set-up. The second rule requests the robot to wait for

a *BenchmarkState* message. It is supposed to start with testing the functionality as soon as the state received equals RUNNING. This allows the RoCKIn partners to set-up any elements necessary for the benchmark, without the possibility for a team to change anything during benchmark execution. The necessity to change elements during the run will be explained for each benchmark in the following sections. Other than in the task benchmarks, the third rule requires the robot to send the benchmarking data online to the CFH as soon as it is available. Specifically, the robot must send a message of type *BenchmarkFeedback* with the required data to the CFH. The robot should do this until the state variable of the *BenchmarkState* messages changes from RUNNING to STOPPED. The functionality benchmark ends when the state variable of the *BenchmarkState* message changes to FINISHED. The strong focus on online communication through the CFH guarantees a fair execution of the benchmark and less chance for error, e.g. as caused by human benchmark operators failing to switch parts in time.

#### 4.1. Object perception

This functionality benchmark has the objective of assessing the capabilities of a robot in processing sensor data to extract information about observed objects. Objects presented to the robot in this functionality benchmark are chosen to be representative for the type of factory scenario that RoCKIn@Work is based on. Teams are provided with a list of individual objects (*object instances*), subdivided into object classes as described in Ref. [3]. The benchmark requires the robot, upon presentation of objects from such a list, to detect their presence and to estimate their class, identity and location. For example, when presented a segment of a T-section metal profile, the robot has to detect that it sees a profile (*class*), with a T-shaped section (*instance*) and its position with respect to the known benchmark set-up reference frame.

##### 4.1.1. Functionality description

The objects that the robot is required to perceive are positioned, one at the time, on a table located directly in front of the robot. Depending on the set-up, this table can either be a separate table outside of the testbed or a workstation within the testbed. The poses of the objects presented to the robot are unknown until they are actually set on the table. For each object presented to the robot, it has to show performance in three distinctive areas as follows:

- Object detection
- Object recognition
- Object localization

The object detection part tests the robot's ability to perceive the presence of an object on the table and associate it to one of the object classes. Object recognition tests the ability to associate the perceived object with a particular object instance within the selected object class. Object localization tests the ability to estimate the 3D pose of the perceived object with respect to the surface of the table. **Figure 7** shows different objects mounted on small wooden plates





**Figure 7.** Objects used during the benchmark. The plate in the foreground is used to acquire the ground truth data.

which fit to the plate in the foreground in only one way. This allows to easily capture the ground truth data.

Feature variation for this functionality benchmark consists only of the variations given by the test itself: The variation space for object features is defined by the (known) set of objects the robot may be exposed to, and the variation space for object locations is defined by the surface of the benchmarking area where objects are to be located.

#### 4.1.2. Procedures and rules

The concrete set of objects presented to the robot during the execution of the functionality benchmark is a subset of a larger set of available objects (*object instances*). Object instances are categorized into classes of objects that have one or more properties in common (*object classes*). Objects of the same class share one or more properties, not necessarily related to their geometry (for instance, a class may include objects that share their application domain). Each object instance and each object class are assigned a unique ID. All object instances and classes are known to the teams before the benchmark, but a team does not know which particular object



instance will be presented to the robot during the benchmark. More precisely, a team is provided with the following information:

- Descriptions/models of all the object instances in the form of 3D textured models.
- Categorization of the object instances into object classes (for instance: profiles, screws and joints)
- Reference systems associated with the table surface and each object instance (for expressing object poses)

Object descriptions are expressed according to widely accepted representations and well in advance of competitions.

During the execution of the task, the robot should perform the task autonomously and without any additional input. The functionality benchmark is carried out by performing the following steps:

1. An object of unknown class and unknown instance is placed in front of the robot.
2. The robot determines the object's class, the instance within that class, and the 3D pose of the object, saving it in the required format.
3. The preceding steps are repeated until time runs out or 10 objects have been processed.

Since this test does not include a dynamic set-up and only a single functionality is tested, teams do not have to consider possible changes in the environment, e.g. a second robot presenting an obstacle for robot motion or changes of the lighting conditions.

#### 4.1.3. Scoring and ranking

Evaluation of a robot's performance in this functionality benchmark is based on

- The number and percentage of correctly classified objects.
- The number and percentage of correctly identified objects.
- Pose errors for correctly identified objects as measured by the ground truth system.
- Execution time (if less than the maximum allowed for the benchmark).

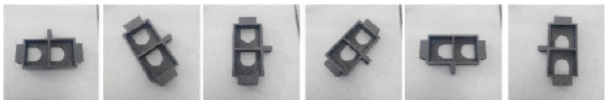
As this functionality benchmark focuses on object recognition, the previous criteria are applied in order of importance. The first criterion is applied first and teams are scored according to their accuracy. Ties are broken by the second criterion, which still applies accuracy metrics. Finally, position error is evaluated as well. Since the position error is highly affected by the precision of the ground truth system, a set of *distance classes* is used. Remaining cases of ties are resolved by execution time.

4.2. Manipulation

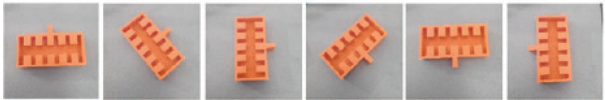
This functionality benchmark assesses the robot’s ability to grasp different objects. An object from a known set of objects is presented to the robot. After identifying the object, the robot needs to perform the grasping motion, lift the object and notify the CFH that it has grasped the object.

4.2.1. Functionality description

The robot is placed in front of the test area, a planar surface. A single object is placed in the test area and the robot has to identify the object and move its end effector on top of it. Then the



Aid tray



File card box



Bearing box A



Bearing box B



Bearing



Axis



Motor with gearbox

Figure 8. Object placement for the manipulation functionality benchmark.

robot should perform the grasping motion and notify that it has grasped the object. The task is repeated with different objects. So far, the following list of classes and instances of objects was used in the manipulation functionality benchmark:

- Containers
  - Assembly aid tray
  - File card box
  - Cover plates
- Bearing boxes
  - Bearing box type A
  - Bearing box type B
- Transmission parts
  - Bearing
  - Motor with gearbox
  - Axis

The objects used in the benchmark are selected from the complete set of parts used in the competition. The precise position of the objects differs in each test (examples are shown in **Figure 8**), which is necessary to avoid that grasping motions can be pre-planned by the teams and to ensure that the grasping motion really depends on the object presented. This test extends the object perception test by a manipulation part.

#### *4.2.2. Procedures and rules*

Teams are provided with the following information:

- The list of objects used in the functionality benchmark.
- Possible placements for each object used in the functionality benchmark.

During execution of the task, the robot should perform the task autonomously and without any additional input. The functionality benchmark is carried out by performing the following steps:

1. An object of unknown class and unknown instance is placed in the test area in front of the robot.
2. The robot determines the correct object class and object instance.
3. The robot grasps and lifts the object, then notifies the CFH that grasping has been performed.
4. The robot keeps the grip for a given time while the referee verifies the lifting.
5. The preceding steps are repeated with different objects.

For each object presented, the robot has to produce the result data consisting of the object's class name and instance name.

As this functionality benchmark does not foresee a dynamic set-up and only a single functionality is tested, teams do not have to consider possible changes in the environment, e.g. a second robot crossing its path or changes of the lighting conditions.

#### 4.2.3. *Scoring and ranking*

Evaluation of a robot's performance in this functionality benchmark is based on

- The number and percentage of correctly identified objects.
- The number and percentage of correctly grasped objects; a grasp is considered successful when the object has no contact with the table any more.
- Execution time (if less than the maximum allowed for the benchmark).

Since this functionality benchmark focuses on manipulation, scoring of teams is based on the number of correctly grasped objects. A correct grasp is defined as the object being lifted from the table such that it is possible for the judge to pass his hand below it. For a grasp to be correct, the position has to be kept for at least 5 seconds from the time the judge has passed the hand below the object. The time the judge needs to verify the lifting of the object takes up to 10 seconds. In case of ties, the overall execution time is considered.

### 4.3. Control

This functionality benchmark assesses the robot's ability to control the manipulator motion and, if necessary, also the mobile platform motion, in a continuous path control problem. The ability to perform this functionality is essential in practice for precise object placement or for following a given trajectory in common industrial applications like welding or gluing. A path (or even a trajectory) is given to the robot. The robot has to follow this path with an end effector on its manipulator (examples shown in **Figure 9**).

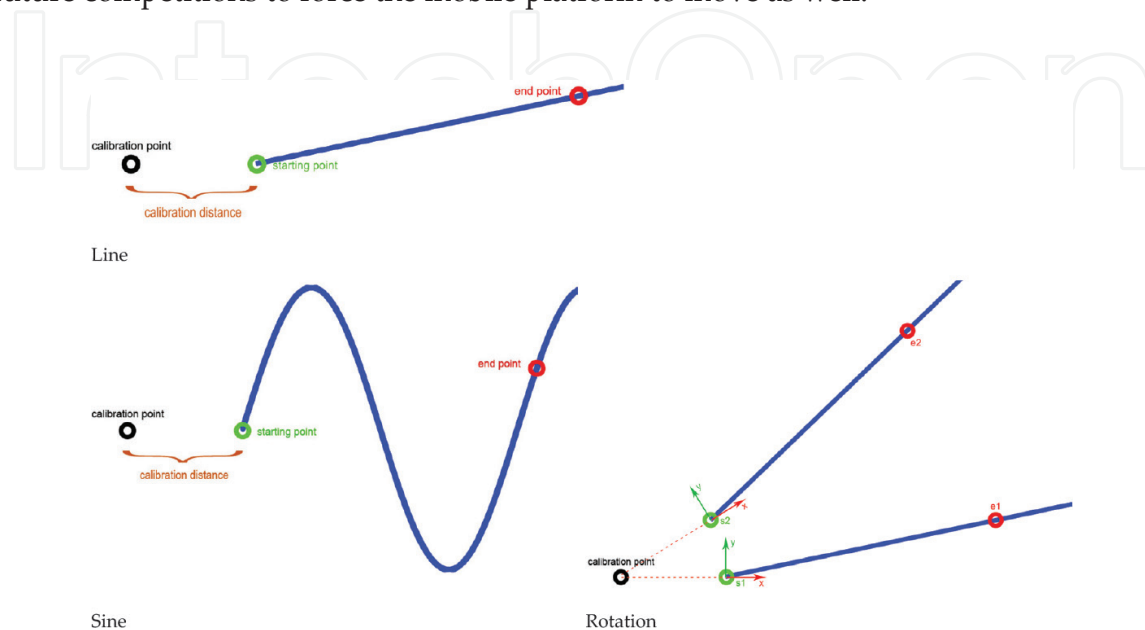
The path is displayed on the table including a reference system. The external ground truth system measures the deviation of the path planned and executed by the robot from the given path by tracking a set of markers attached to the end effector.

#### 4.3.1. *Functionality description*

The robot is placed in front of the test area, a planar surface. It first places its end effector on the top of a calibration point, then on the starting point with a fixed offset from the calibration point. At each of the two points, a manual calibration is performed by adjusting the position of the printed path (the table or sheet of paper). In order to synchronize the reference frames of the robot and the ground truth system, the robot detects the reference and starting point and then notifies the ground truth system about the positions of those points. The robot starts to follow the path and reports this to the CFH. After it finishes the movement, it has to signal this to the CFH.



Possible feature variations are the different paths the robot has to follow. In the second competition, where this benchmark was introduced first, the path was a simple line and sine. In future competitions, the path could be extended to become a general spline and it could be specified as trajectory including required velocity and acceleration vectors. The path is currently limited to the manipulator workspace, but can be extended well beyond this workspace in future competitions to force the mobile platform to move as well.



**Figure 9.** Example paths the robot had to follow.

#### 4.3.2. Procedures and rules

Teams are provided with the following information:

- A mathematical description of a line in two-dimensional (2D) space.
- A mathematical description of a sine in 2D space.
- A list of generated points for both line and sine.
- Just before a competition run, the selection of the line or the sine for each run is published.

The path is provided including a starting point and a reference point next to it to enable calibration and synchronization with the ground truth system. Note that this task is not executed with a feedback from any vision sensor from the team, but only tests a pre-planned path and the online continuous path control ability of the robot!

During the execution of the task, the robot should perform the task autonomously and without any additional input. The functionality benchmark is carried out by executing the following steps:

1. The robot/team is provided with the selection of the specific path in advance.
2. The robot moves to the defined reference point within his coordinate system. Manual adjustment is then performed by a referee: the paper with the printed path is placed under

the actual position of the robot's end effector. (This is mainly important for the audience to get a visual feedback and to see the predefined path).

3. The robot moves to the defined starting point of the path, which is defined a few centimetres away with respect to the reference point. Another manual adjustment of the paper is then performed.
4. The CFH tells the robot when to start moving.
5. The robot moves its end effector along the path until the end point of the path is reached and reports the termination of path execution to the CFH.

#### 4.3.3. Scoring and ranking

Evaluation of a robot's performance in this functionality benchmark is based on

- The overall deviation of the executed from the given path, measured in terms of the areas summing-up between the given and the executed path (constant deviations are eliminated).
- The number of completely executed path movements.
- Execution time (if less than the maximum allowed for the benchmark).

As this functionality benchmark focuses on control, the scoring of teams is based on the size of the area describing the deviation from given and executed path. In case of ties, the overall execution time is considered.

## 5. Summary

This chapter provides detailed information on the RoCKIn@Work competition. First, the competition, the concepts that build its foundation, and the intentions behind it are explained. After that, elements for building an open domain testbed for a robot competition set in the industrial domain are introduced and the most important aspects of benchmarking in competitions are outlined. The main part of this chapter covers in detail the three task benchmarks, *Prepare Assembly Aid Tray for Force Fitting*, *Plate Drilling* and *Fill a Box with Parts for Manual Assembly*, as well as the three functionality benchmarks, *Object Perception*, *Manipulation* and *Control*.

## Author details

Rainer Bischoff<sup>1</sup>, Tim Friedrich<sup>1\*</sup>, Gerhard K. Kraetzschmar<sup>2</sup>, Sven Schneider<sup>2</sup> and Nico Hochgeschwender<sup>2</sup>

\*Address all correspondence to: tim.friedrich@kuka.com

1 KUKA Roboter GmbH, Germany

2 Bonn-Rhein-Sieg University of Applied Sciences, Germany

## References

- [1] Niemueller T, Zug S, Schneider S, Karras U. Ubbo Visser, Gerald Steinbauer, Alexander Ferrein. Knowledge-based instrumentation and control for competitive industry-inspired robotic domains. In: *Künstliche Intelligenz*. Springer Berlin Heidelberg. 30th ed. 2016. pp. 289-299
- [2] Schneider S, Hegger F, Hochgeschwender N, Dwiputra R, Moriarty A, Berghofer J, Kraetzschmar G. Design and development of a benchmarking testbed for the factory of the future. In: *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA): Special Session on Mobile Robotics in the Factory of the Future*; 8-11 September 2015; Luxembourg. IEEE; 2015
- [3] RoCKIn Project. Project Website [Internet]. 2014. Available from: <http://rockinrobotchallenge.eu/> [Accessed: 26 May 2017]
- [4] Ahmad A, Awaad I, Amigoni F, Berghofer J, Bischoff R, Bonarini A, Dwiputra R, Fontana G, Hegger F, Hochgeschwender N, Iocchi L, Kraetzschmar G, Lima PU, Matteucci M, Nardi D, Schiaffionati V, Schneider S. RoCKIn Project D1.2 "General Evaluation Criteria, Modules and Metrics for Benchmarking through Competitions". 2014. Available from: [http://rockinrobotchallenge.eu/rockin\\_d1.2.pdf](http://rockinrobotchallenge.eu/rockin_d1.2.pdf) [Accessed: 26 May 2017]
- [5] Amigoni F, Bonarini A, Fontana G, Matteucci M, Schaffionati V. To what extent are competitions experiments? A critical view. In: *Workshop on Epistemological Issues in Robotics Research and Research Result Evaluation*; Hong Kong. ICRA 2014. 05 June 2014. <http://rockinrobotchallenge.eu/publications.php>

IntechOpen